Lua: a linguagem de script do NCL

2020

Prof. Douglas Paulo de Mattos

Linguagem de Script

TecGraf (PUC-Rio) & Petrobrás

Usada ha anos pela indústria de jogos

Interpretada

Tipagem dinâmica

Gerência automática de memória

Facilidade de estruturação de dados

Facilidade para manipulação de String

Segura

Facilidade para comunicação entre os componentes

Integração com NCL

Aumenta a expressividade

Permite inserir objetos imperativos no documento NCL

Inserido com o elemento <*media*> cujo conteúdo (*src*) é o código Lua

Atributo *type* usado para indicar o objeto imperativo Lua

Integração com NCL

Elemento <media type="application/x-ncl-NCLua" src="caminho do código Lua">

Também possui *area* (trecho do código - função indicado no *label*) e *property* (variáveis do código)

Programador decide o comportamento das âncoras

```
<media id="objetoLua" src="exemplo.lua">
      <property name="propriedade1"/>
      <property name="propriedade2" value="0"/>
      <area id="ancora1" label="final"/>
      <area id="ancora2"/>
</media>
```

Start nó Lua -> executa todo o código (ancora total padrão)

Integração com NCL feita através de módulos

Event

Canvas

Settings

Persistent

Integração com NCL

Elos NCL podem iniciar ou parar um código Lua

Código Lua pode iniciar ou parar suas âncoras

Integração através de módulos

Módulo **Event**

Permite que objetos NCLua se comuniquem com o documento NCL e outras entidades externas (tais como controle remoto e canal de interatividade)

Módulo **Canvas**

Oferece funcionalidades para desenhar objetos gráficos na região do NCLua

Módulo **Settings**

Oferece acesso às variáveis definidas no objeto *settings* do documento NCL (objeto do tipo “application/x-ncl-settings”)

Módulo **Persistent**

persistência de dados da aplicação

cria variáveis que podem ser recuperadas para uso futuro

exporta uma tabela com variáveis persistentes
entre execuções de objetos imperativos

Para receber eventos externos o código Lua deve

- Registrar pelo menos uma função de tratamento: **`event.register`**
- Obs.: o nome da função a ser registrada pode ser qualquer um.

Estrutura comum a todos os scripts:

```
...                                  -- código de iniciação
function tratador (evt)
      ...                        -- código de um tratador
end
event.register(tratador) -- registro de pelo menos um tratador
```

O parâmetro evt é definido pela ação (start/end) ocorrida no NCL no nó imperativo

Obs.:

- A **definição** do tratador e seu registro são inicializador antes de qualquer evento.

- Apenas o **código do tratador** é chamado toda vez que ocorre um evento externo.

Evento é representado por:

```
evt = {
  class  = 'key',
  type   = 'press',
  key    = 'RED',
}
```

- Principais classes
  - Classe **ncl**: usada na comunicação entre um NCLua e o documento NCL que contém o objeto de mídia.
  - Classe **key**: representa o pressionamento de teclas do controle remoto pelo usuário.

Evento de post - sinaliza seu estado ao doc NCL:

```
event.post {
    class  = 'ncl',
    type   = 'presentation',
    action = 'stop',
}
```

Obs.:

- A função de envio de eventos nunca aguarda o retorno de um valor.
- Caso o destino necessite retornar uma informação ao NCLua, o fará através do envio de um novo evento.

Exemplo de elo NCL iniciando um código Lua

arquivo NCL que contém o objeto NCLua

```
<link xconnector="onBeginStart">
    <bind role="onBegin" component="videoId"/>
    <bind role="start"   component="luaId"/>
</link>
```

arquivo NCLua

```
-- Evento recebido pelo tratador do NCLua no
-- disparo do elo:
evt = {
    class  = 'ncl',
    type   = 'presentation',
    action = 'start',
}
```

Exemplo código Lua disparando evento no doc NCL

arquivo NCL que contém o objeto NCLua

```
<link xconnector="onBeginStart">
    <bind role="onEnd" component="luaId"/>
    <bind role="start" component="imagemId"/>
</link>
```

arquivo NCLua

```
-- O elo acima será disparado quando o evento a seguir
-- for postado pelo NCLua 'luaId':
event.post {
    class  = 'ncl',
    type   = 'presentation',
    action = 'stop',
}
```

Observações

Lua vazio nunca termina - não há tratador de evento

```
-- 1.lua
-- vazio
```

O que esse script implementa?

```
-- 2.lua:
function tratador (evt)
  if (evt.class == 'ncl') and (evt.type ==
'presentation')
     and (evt.action == 'start') then
    event.post {
        class  = 'ncl',
        type   = 'presentation',
        action = 'stop'
    }
  end
end
event.register(tratador)
```